



OMNi
bnk

**Exploración De Datos
Con Neo4j**

Content

- 03 Some definitions
- 08 Why do we use graph databases?
- 03 What is Neo4J?
- 04 Exploring data with python

Juan Salvador Bahamón Arias
Data scientist / Omnibnk
Software Engineer / Universidad Distrital Francisco José de Caldas
jbahamon.arias@gmail.com
[github/jusalbari/](https://github.com/jusalbari/)

A woman with curly hair, wearing a light-colored blazer over a striped shirt, is smiling and working on a laptop. She is in a clothing store, with racks of clothes and a mannequin visible in the background. The image has a purple and orange color overlay.

Some definitions

What is a database?

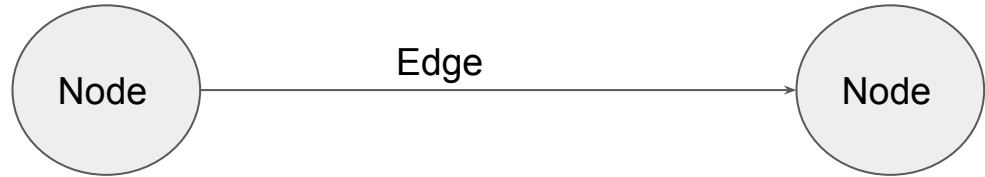
It's a data collection

ORACLE



What is a graph?

It's a structure that represent with points and lines the relationships between a couple of elements



What is a graph database?



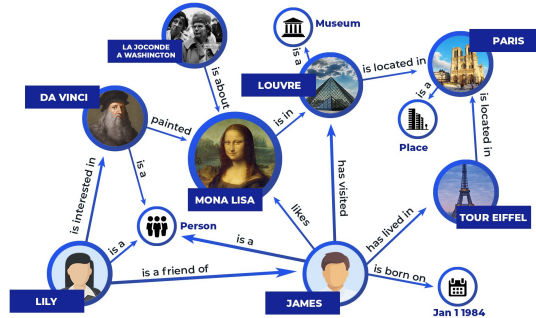
It's a collection of data that uses graph structures for semantic queries with nodes, edges, and properties to represent and store data.

Use cases with graph databases

Fraud Detection

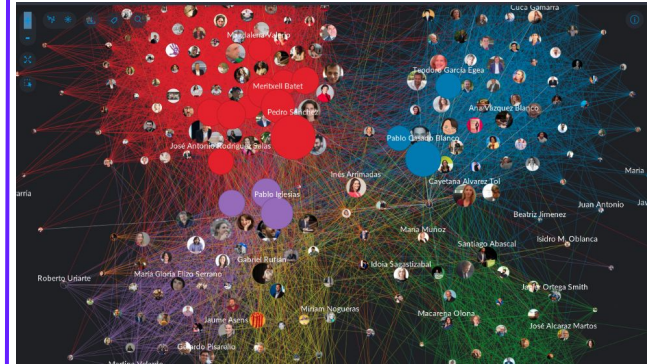


Knowledges Graphs



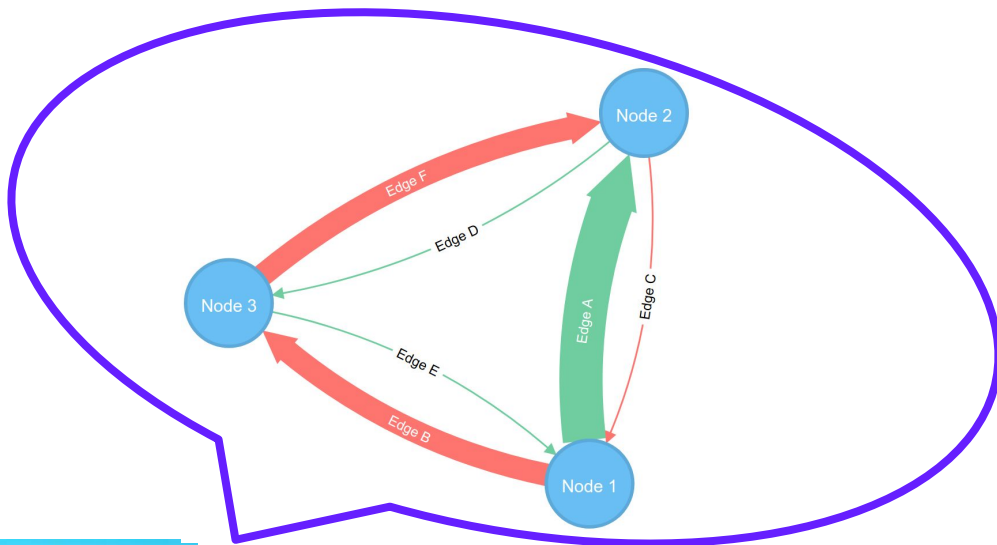
<https://yashueth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kqa/>

Social Networking



https://elpais.com/tecnologia/2019/11/14/actualidad/1573769942_710891.html

What is Neo4j?



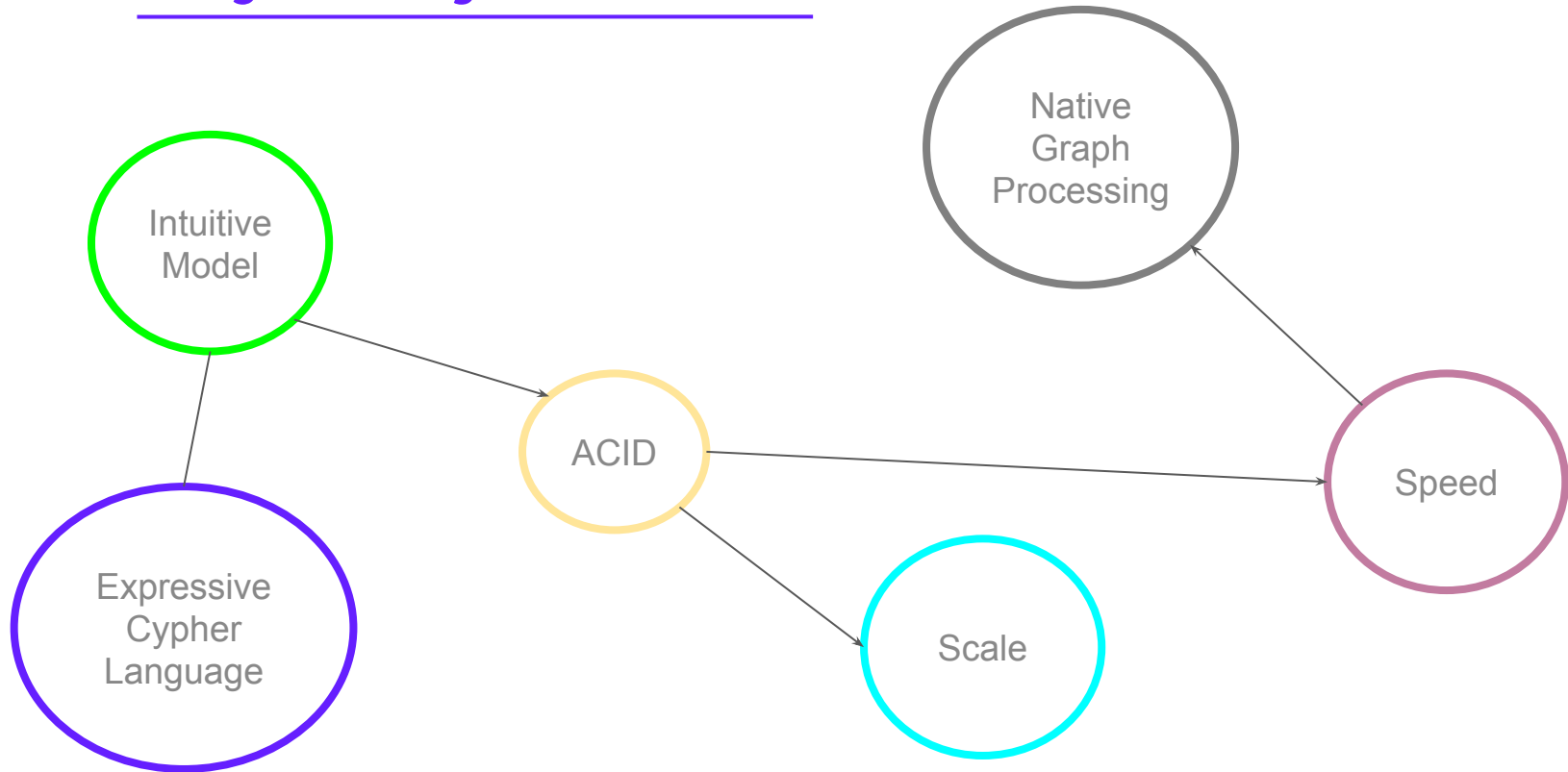
Since 2010
Language Java
Current version is 4.0.0
Open Source



“Unlike other databases, Neo4j connects data as it is stored, enabling it to traverse connections orders-of-magnitude faster”.

[\[https://neo4j.com/neo4j-graph-database\]](https://neo4j.com/neo4j-graph-database)

Why Neo4j?



A woman with dark hair in a bun, wearing large over-ear headphones and a grey button-down shirt, is seated at a long wooden desk in a modern office. She is smiling and looking towards the right. Her hands are on a laptop keyboard. The desk is cluttered with various items including a red mug, a small red container, and some papers. In the background, other people are working at similar desks, and the office has a bright, open feel with large windows. The image is overlaid with a blue-to-orange gradient.

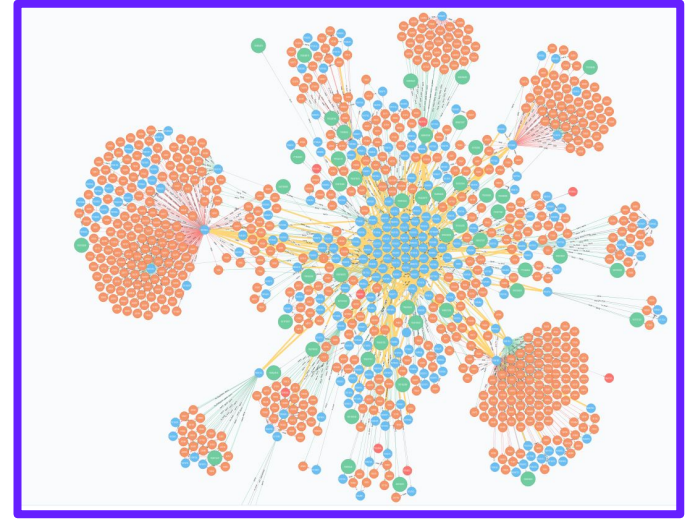
Let's Practice

Github

<https://github.com/jusalbari/pycon2020>

Let's Practice

- Install Neo4j
- Install Python driver Py2neo
- Create database (nodes and edges)
- Queries structure
- Exploring data with Python



Install Ne4j

Install with Docker

```
sudo apt-get update
```

```
sudo apt-get remove docker docker-engine  
docker.io
```

```
sudo apt install docker.io
```

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

Command

```
docker run --publish=7474:7474 --publish=7687:7687  
--volume=$HOME/neo4j/data:/data --volume  
$HOME/neo4j/import:/var/lib/neo4j/import  
neo4j
```

Default: user: neo4j // password: neo4j

Install on your pc

Opcion 1 - tar

<https://neo4j.com/download-center/#community>

```
tar -xf <filecode>.
```

```
cd neo4j-community-4.0.0/
```

```
bin/neo4j console
```

Opcion 2 - application

<https://neo4j.com/docs/operations-manual/current/installation/>

Install OpenJDK

<https://mkyong.com/java/how-to-install-java-jdk-on-ubuntu-linux/>

Install py2neo, pandas and numpy

Opcion - pip

```
pip install py2neo  
pip install pandas  
pip install numpy
```

Opcion - conda

```
conda install -c conda-forge py2neo  
conda install -c anaconda pandas  
conda install -c anaconda numpy
```

Implementation in python

Import packages

```
import pandas as pd  
from py2neo import Graph, Node, Relationship
```

Create Graph connection

```
graph = Graph(uri='bolt://localhost:7687',user='neo4j', password='neo4j')  
tx = graph.begin()
```


Create nodes and edges (OGM)

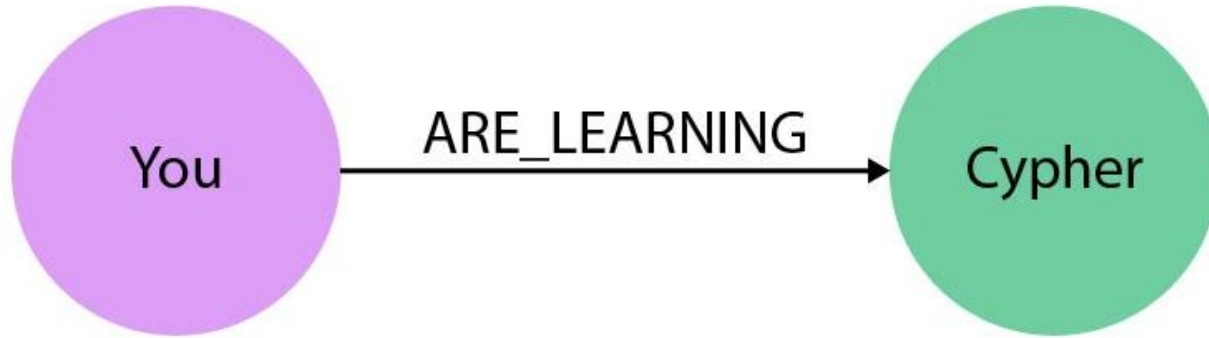
Create Nodes

```
node1 = Node("Node_type_1", name="Node 1", attr_2="value_attr2")
node2 = Node("Node_type_1", name="Node 2")
node3 = Node("Node_type_1", name="Node 3", attr_4=3 , attr_n=)
node4 = Node("Node_type_1", name="Node 4", attr4=23)
```

Create Edges

```
edge_a = Relationship(node1, "Edge A", node2)
edge_b = Relationship(node1, "Edge B", node3)
edge_c = Relationship(node2, "Edge C", node1)
edge_d = Relationship(node2, "Edge D", node3)
edge_e = Relationship(node3, "Edge E", node1)
edge_f = Relationship(node3, "Edge F", node2)
```

Cypher, the Graph Query Language



Match

```
Match (p:Person)-[:ARE_LEARNING]->(l:Language)
```

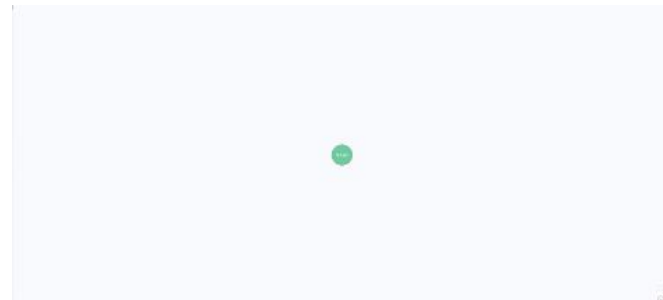
Match statement

Match option 1

```
Match (p:Person)-[:ARE_LEARNING]->(l:Language {name:'Cypher'})  
return p, l
```

Match option 2

```
Match (p:Person)-[:ARE_LEARNING]->(l:Language)  
return p.name, l.name
```



Match option 3

```
Match (p:Person)-[:ARE_LEARNING]->(l:Language)  
return p, l.name
```



OMNi
bnk

GRACIAS